Noise Level Calculator for Echo Canceller

Field Of The Invention

The present invention relates generally to telecommunications systems, and in particular to a noise level calculator for detecting noise in a telephone line echo canceller.

Background Of the Invention

It is known in the design of line echo cancellers which utilize adaptive filters to incorporate a non-linear processor (NLP) for removal of residual echo signals (e.g. due to non-linearity, distortion, or added signal noise). In order to avoid noise switching being heard on the far end side, it is important that the noise level of the signal applied to the NLP be calculated to distinguish between noise and residual echo.

Noise level calculation is also useful in determining if the reference signal applied to the adaptive filter is noise or a non-noise segment. If noise is detected, then updating of the filter coefficients may be suppressed.

Summary Of The Invention

According to the present invention, a noise level calculator is provided for monitoring the noise level of the error signal applied to the NLP and the noise level of the reference signal applied to the adaptive filter, in order to accomplish the objects set forth above. In contrast with prior art noise detectors which track not only the noise segments but also the signal level and conclude that the noise level is directly proportional to the lowest accumulated signal energy, the noise level calculator of the present invention uses the variance in the signal energy to determine background

noise level. Consequently, the noise level calculator of the present invention actually locates the noise periods and adapts to changes in the variance of noise energy during these periods.

Brief Description Of The Drawings

A preferred embodiment of the present invention will now be described in greater detail with reference to the following drawings, in which:

Figure 1 is a block diagram of a line echo canceller employing the noise level calculator of the present invention; and

Figure 2 is a flowchart showing steps for implementing the noise level calculator according to the preferred embodiment.

Detailed Description Of The Preferred Embodiment

With reference to Figure 1, a Line Echo Canceller (LEC) is shown for canceling echo signals from a line echo path (between **Sin-line** and **Rin_line**). An adaptive filter algorithm (typically the well known adaptive LMS algorithm) is implemented within control logic block 9 to perform the echo cancellation function. The adaptive filter output is subtracted within summation block 3 from the input line signal (**Sin-line**) to create an error signal (**ein**). . As discussed above, NLP 5 is provided for removal of residual echo signals due to non-linearity, distortion, added signal noise, etc. A double talk detector 11 is included for disabling the NLP 5 during a double talk condition (i.e. when the near-end party begins talking, in which case the signal becomes near-end speech plus far-end echo).

According to the present invention, a noise level calculator 13 is provided for continually monitoring the noise level for the error signal **ein** as well as for the

reference signal **Rin-line**. The noise level of the error signal is used by the NLP component 5 to decide if the sample is noise or residual echo. If noise is detected, it is transmitted as is but if residual echo is detected, the NLP 5 generates a noise sample. The noise level of the reference signal is used by the adaptive filter algorithm in control block 9 to decide if the signal is speech or noise. If it is noise, the echo canceller coefficients are not updated.

Turning now to Figure 2, a preferred algorithm is shown for implementing the noise level calculator of the present invention. The equations in Figure 2 are written for optimal performance in a C compiler. Two inputs are required for the implementation, namely an input signal, **xin** and a tone decision parameter, **tone_decision**. The internal state of the noise level calculator and its local variables is held in a local workspace, (i.e. the noise level structure set forth below).

The same function is used to calculate the noise level for the reference signal, **Rin_line**, and the error signal, **ein**, of the LEC in Figure 1 The input sample, **xin**, for each noise calculator implementation is chosen to be either **ein** or **Rin_line**.

The Noise Level Structure and the required local variables for each such implementation are as follows, with reference to Figure 2:

Structure (*NoiseLevel):

(*NoiseLevel).count: this is a counter for the sample window

(*NoiseLevel).level: this represents the NoiseLevel

(*NoiseLevel).accum[0]: current accumulation of the input signal

(*NoiseLevel).accum[1]: previous accumulation of the input signal

(*NoiseLevel).variance: variance of the accumulation

(*NoiseLevel).flag_no_update: invalid window flag, Noise Level is not updated

xin: input sample

diff: difference between the previous and current accumulations

tone_decision: flag is 1 if signal is a Tone

nlevel_count: number of accumulations before the Noise Level is updated

The algorithm of the present invention is based on the assumption that the energy variance of a noise segment is much lower than the energy variance of a voice segment. After determining that no tone is present (step A), samples within a window of 256 samples (32 msec) are accumulated (i.e. the signal energy within the window is calculated). (**(*NoiseLevel).accum[0] = (*NoiseLevel).accum[0]+abs(xin)** in step B). When the window is completed (**(NoiseLevel.count>Window_size ?** in step D), the noise level is updated and the result is saved in memory. It is important that the sample accumulation does not exceed a maximum level (**limit (*NoiseLevel).accum[0] to Max_limit**, in step C) in order to ensure that the variance and noise level calculations do not become corrupted. In the event that the accumulation is invalid (**Yes (accum is invalid)**, in step D), the Noise Level is not evaluated for the next two windows (as a result of the flag being set at **(*NoiseLevel).flag_no_update = 2** in step C and then twice decremented **(*NoiseLevel.flag_no_update-=1** in step D). Two windows are considered invalid at start up as well as when the Max-limit has been reached. They are not used, so as to fill in or clear the history of accumulations respectively.

When two valid accumulations of samples are available, the difference is calculated, which is then used to update the variance of these accumulations as a weighted average of the difference and the previous values of the variance parameter (i.e. **(*NoiseLevel).variance+= (diff – (*NoiseLevel.variance)>>3)** adjusts the variance parameter to the existing (i.e. previous) variance parameter plus a multiple (÷8) of the difference (diff) minus of the previous variance parameter, and **(*NoiseLevel.variance= diff)** adjusts the variance parameter to the previous variance parameter. Thus, in the embodiment of Figure 2, the variance is calculated/updated with an attack ratio of 1 and a decay ratio of 8 (i.e. >>3 in step E). These attack and decay rates were chosen empirically so that the noise level is not updated during short periods of low energy (e.g. caused by fricatives in a voice segment). Alternatively, other attack and decay ratios may be chosen to suit different applications. Consequently, the attack rate is aggressive and the decay rate is slower than the attack rate.

When the variance decreases to the accumulation level divided by a pre-determined scale factor (**(*NoiseLevel).variance < (*NoiseLevel).accum[0]>>3?** in step E), the current accumulation is considered to bepart of a noise segment. This factor (>>3 = 8 in Step E) was chosen after comparing the ratio between different noise levels and corresponding variances of the accumulations. When it is decided that the accumulation is noise (step F), the noise level is calculated/updated with a decay ratio of 8 (i.e. >>3 in Step F). A decay ratio other than 8 may be chosen for different applications.

In the case where the noise level is greater than the accumulation, the noise level is reset to the current accumulation value (step G). This is to ensure that the noise level calculator is biased towards the lowest possible noise level.

Finally, some variables are re-initialized for the next accumulation window (step H).

It should be noted that, for the **ein** signal, the noise level is multiplied by a factor of 0.013 (which is 3.3/256, 256 being the Window size) before it is compared to an actual sample. For the **Rin_line** signal, the noise level is multiplied by a factor of 0.2 (50/256) which is on the high side for the noise threshold.

Alternatives and variations of the invention are possible. For example, real energy calculations can be used instead of taking the absolute value of the samples in step B, different window sizes may be used, and different attack and decay rates may be specified for updating the variance (step E). Furthermore, it is contemplated that the algorithm of the present invention may also be applied to detect voice (i.e. the absence of noise) and may be applied to the operation of a comfort noise generator for silence suppression. All such alternative embodiments and applications are believed to be within the sphere and scope of the invention as defined by the claims appended hereto.